CSCI 4131 – Internet Programming
Assignment 4

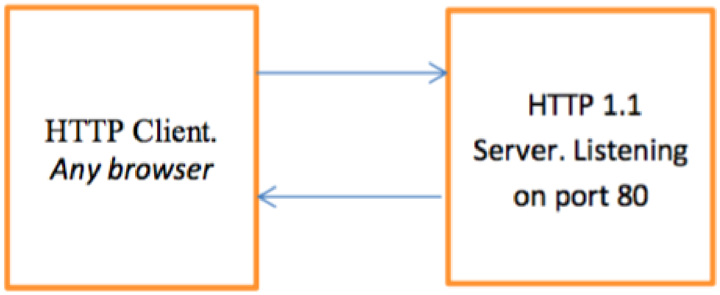
Posted 10/19/2017

Due Sunday, 10/29/2017 at 11:55 PM

1 Description

The objective of this assignment is to learn HTTP protocol (HTTP1.1) and build HTTP server with very limited functionality. Using Python and TCP sockets you will program some of the functionality of an HTTP server for this assignment. You will need to read through RFC 2616 for details on the HTTP 1.1 protocol, in addition to the links to the other resources specified in lecture.

When a web client (such as Google Chrome) connects to a web server (such as ***www.google.com***) the interaction between them happens via the Hyper Text Transfer Protocol (HTTP).



An overview of how an HTTP server functions is as follows:

1. An HTTP client connects to the HTTP-server and makes a HTTP request to request a web resource.
2. The HTTP Sever parses the request header fields. The request can be of type GET, POST, HEAD, etc. For a GET requests, the server identifies the requested resource (for example an HTML file) and checks if the resource exists.
3. The HTTP server then generates an appropriate HTTP response message. If the requested resource is found, the HTTP server includes successful (2xx) status code in the response headers along with other meta data such as the Content Type and Content Length and sends the resource data as the message body. If the requested resource is not found, then the HTTP Server sends response with status code 404.

2 Preparation and Files Provided

The following files are provided for this assignment:

- 403.html: this file should be sent to the client after forbidden header code.
- 404.html: this file should be sent to client when the requested file is not found.
- private.html: this file is the private file that triggers 403 forbidden code.
- calendar.html: replace this file with your own calendar.html file from Assignment 1.

To give above files proper permissions, please execute following commands after downloading the files to your folder or directory

chmod 640 private.html

chmod 644 403.html

chmod 644 404.html

chmod 644 calendar.html

3 Functionality

When started, your server will establish a socket; bind to a port, and listen for connections. You can send a request for your calendar to the server with your web browser using the following URL:

```
http://<host>:<port>/calendar.html.
```

For this assignment, `<host>` will be `localhost` and `<port>` should default to `9001`.

Your code can not use the httplib module of the Python standard library. You should do your own socket programming on this assignment. You can build on the socket level client and server programs that Professor Challou reviewed in lecture.

When invoked with no arguments, your server should bind to port 9001 and serve requests. It should also accept one optional command line argument which specifies the port to bind to. Three example calls to start your server are as follows:

- ***`python myServer.py`***
- ***`python myServer.py –port 9002`***
- **`./myserver.py`**

Note, you should use Python 3 to implement your HTTP server, and your server should log all incoming requests to STDIN.

4 HTTP Protocol

HTTP is a protocol of non-trivial size. We will only be implementing a small, functional subset of HTTP, specifically: GET, HEAD, and POST requests as discussed below

4.1 GET Requests

GET requests are the most commonly encountered requests. When your web browser requests a webpage from a server, it is issuing a GET request. Here is an example GET request that will be received by your server:

```
GET /calendar.html HTTP/1.1
Host: localhost:9001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:33.0)
Gecko/20100101 Firefox/33.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

For our simplified server, only the first line of the request is important. We need to extract the requested URI (calendar.html) and serve it to the client.

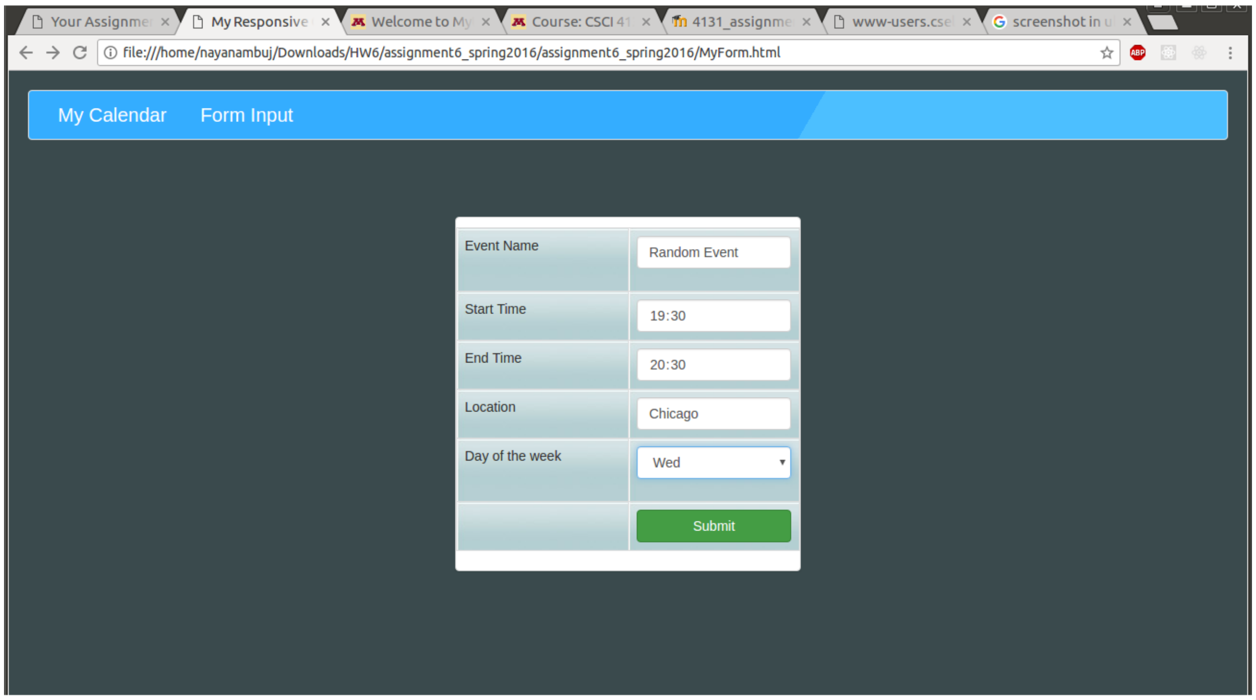
For example, if you enter the following address in your browser – `http://localhost:9001/calendar.html`, the browser should display the calendar file that you developed for the first homework.

4.2 HEAD Requests

HEAD requests are almost identical to GET. Instead of returning a status code followed by the information in the URI, your server should only send the status code (for example HTTP/1.1 200 OK).

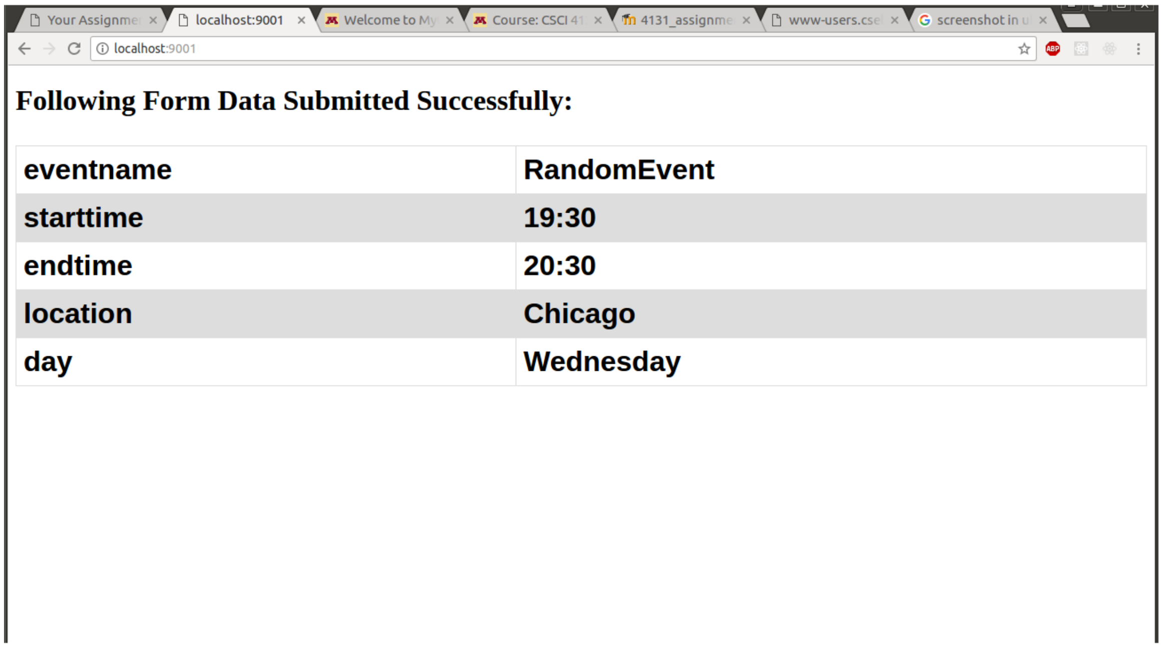
4.4 POST requests

You will use the form that you developed for first homework. In the first homework, the form was submitted to http://www-users.cselabs.umn.edu/~tulaj001/form_handler.php. For this homework, you will instead submit your form to your python HTTP server. You can achieve this by changing the method of the form to "post" and action as http://localhost:9001. See the pictures on the next page for an example.



Event Name	Random Event
Start Time	19:30
End Time	20:30
Location	Chicago
Day of the week	Wed

Upon successful submission of your form (similar to the form pictured above), your server should return an HTML page with all the submitted information as per the screen shot below:



Following Form Data Submitted Successfully:

eventname	RandomEvent
starttime	19:30
endtime	20:30
location	Chicago
day	Wednesday

4.5 Redirection Response

Additionally, you will also send redirection responses for certain URLs. If the request is for a resource named "csumn", then the client should be redirected to the following location:

`https://www.cs.umn.edu/`

For example if the URL `is` `http://comp1.cs.umn.edu:5555/csumn`, then the request should be redirected to: `https://www.cs.umn.edu.` The browser should be automatically redirected to the new URL. For this, your server should send an appropriate response message with required headers. You will have to include appropriate location headers in the response. The redirection response would include the "`Permanently moved`" status code. Please refer to section 10.3 of RFC 2616 for details.

4.6 Responses to error conditions

Your HTTP server should handle error conditions: 403, 404, 405, and 406. It should send appropriate error responses as specified below.

1. If the resource requested does not have appropriate permissions (i.e., it is not world-readable), then the server should send a 403 error response.

2. If the resource requested is not found, the server should send a 404 error response.

3. If the request is anything other than GET, HEAD, OR, POST, (note, the methods must be capitalized, as specified in the HTTP protocol), then the server should send a 405 error response (method not allowed).

4. If the request contains accept headers, and the content characteristics of the requested resource requested are not acceptable according to the accept headers, your server should send a 406 error response. For example, if the accept headers in a GET request specify that html files should be returned, and the requested entity is an image file, the server should send a 406 the error response.

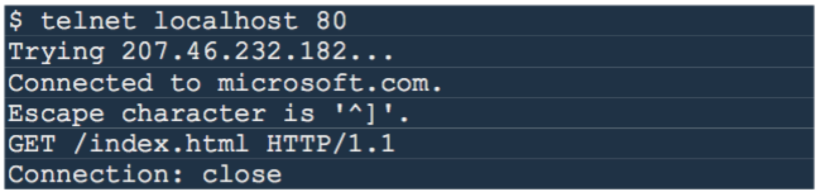
Note, your server should also include an appropriate error message in the response body, specific to the error condition. If we do not provide an html file for the error code, include an appropriate error message plain text in the response to the client (i.e., Browser, Telnet, curl, etc.).

5.Testing Guidelines

To run your HTTP Server, you should pick a port number above 5000. You can test the HTTP server using an **HTTP client, a browser** (`e.g., chrome or Firefox`), **the POSTMAN utility** (`https://www.getpostman.com/),` **curl** (`https://curl.haxx.se/docs/httpscripting.html`), or **telnet** as follows:

a) Suppose you are running the server on a machine named "silver.cs.umn.edu" and on port number 5555, then your URL would be http://silver.cs.umn.edu:5555/<resource-path>.

b) Use telnet in the linux terminal (`http://www.esqsoft.com/examples/troubleshooting-http-using-telnet.htm`). Eg:



```
$ telnet localhost 80
Trying 207.46.232.182...
Connected to microsoft.com.
Escape character is '^]'.
GET /index.html HTTP/1.1
Connection: close
```

c) Write your own python HTTP client

If you run the HTTP server on any of the CSE lab machines, then you will not be able to connect to your server from any machine outside the CSE domain due to firewall. Therefore, you should run both the client (or browser) and server on CSE machines only. You can develop and your HTTP server and any client on your home machine, but you MUST test it on the CSE LABS machines.

NOTE: All assignments will be graded on the cselabs machines - so make sure to test your server on a cselabs machine to ensure it functions correctly as specified in the evaluation criterial below. If your server does not function correctly, or function at all on the cselabs mahines, it will be graded accordingly.

6 Submission Instructions

• Submit your server program that is renamed to <UMN x500> myServer.py

User *john1234* should submit *john1234_server.py*

• You don't need to provide any extra files. We will copy our own 403.html, 404.html, private.html, calendar.html, form.html, etc. when we test your code.

PLEASE ENSURE TO TEST YOUR CODE ON CS LAB MACHINES.

7 Evaluation Criteria

Your submission will be graded out of 100 points (+ 20 Bonus Points) on the following items:

• Server establishes a socket and binds to 9001 by default. 5 **points**

• Server accepts a parameter to change port. 5 **points**

• Server accepts connections from clients and reads incoming messages. **10 points**

• Server correctly identifies GET requests, HEAD requests. **15 points**

• Server correctly processes POST and responds to POST request. **20 points**

• Server correctly responds with 200 and serves requested page (GET and HEAD). **15 points**

• Server correctly responds with 406. **10 points** (Bonus)

• Server correctly responds with 405. **5 points**

• Server correctly responds with 403. **5 points**

• Server correctly responds with 404. **5 points**

• Server redirection 301. **10 points** (Bonus)

• Server logs requests to STDIN. 5 **points**

• Source code is documented and readable. **5 points**

• Submission instructions are followed. **5 points**

Reminder: All assignments will be graded on the cselabs machines - so make sure to test your server on a cselabs machine to ensure it functions correctly as specified in the evaluation criterial above.